Towards a Type-Logical HPSG

MILTIADIS KOKKONIDIS
University of Oxford
miltiadis.kokkonidis@clg.ox.ac.uk

ABSTRACT. Type-Logical Head-driven Phrase Structure Grammar[1] is a new proof-theoretic constraint-based theoretical framework based on Linear Logic (Girard 1987). TL-HPSG places the core aspects of HPSG (Pollard and Sag 1994) on a setting inspired by Glue (Dalrymple 1999) and in particular its recent first-order formulation (Kokkonidis 2008). The move to such a formal setting brings HPSG closer to Type-Logical Categorial Grammar (Morrill 1994; Moortgat 1997) without betraying the core intuitions of HPSG. At the same time, it forces reflection on some of the design decisions in the original framework and a comparison with its type-logical counterpart. For one, the most fundamental principles and generalisations of HPSG come without stipulation in the new type-theoretic setting. Also, given that the encoding employed does not require the use of the '!' linear-logic modality, decidability follows.

1 Introduction

It was the realisation that the move to First-Order Glue (Kokkonidis 2008) and its underspecification-based version (Kokkonidis 2006) in particular allowed the encoding of feature structures used in Head-driven Phrase Structure Grammar that lead to the development of their type-theoretic versions discussed here. The role of Glue is that of the intermediary between syntax and semantics; this was its role in Lexical Functional Grammar for which it was originally invented (Dalrymple, Lamping, and Saraswat 1993) but also the role Asudeh and Crouch (2002) envisaged for it to have in HPSG. Originally Glue types could only be anchored to hierarchical feature structures, but the recent aforementioned developments gave it the capability to encode such structures. This lead to the question of whether or not it could be the syntax-semantics interface type system itself that constructs these structures while also enforcing any linear-precedence constraints. Indeed it can. The underspecification-based $G3_i$ type system of Kokkonidis (2006) fits the intended use like a glove. As for linear-precedence constraints, all that is needed is a concept of a span for them to be captured while remaining in a commutative type-logical setting (Kokkonidis 2007a). So rather than be used as part of HPSG (cf. Asudeh and Crouch 2002), Glue can now be the underlying foundation for a type logical version of HPSG.

[1] TL-HPSG is not to be confused with Higher-Order Grammar, presented on occasion as a type-logical HPSG (Pollard 2004). Pollard now pursues a program of developing Convergent Grammar (CVG) that will be interesting to compare at some point with TL-HPSG although this will not be currently pursued.

2 An example

Even a very simple example can help reveal the differences between HPSG and its proposed type-logical version. The two lexical entries below suffice to parse sentence (3). There is no need for rules and principles beyond the very simple logic used as the formal foundation of TL-HPSG. It is worth noticing also that, in comparison to HPSG, the feature structures are significantly more compact as they are missing the semantic and valence components. Finally, nothing special has to be said about the syntax-semantics interface; since TL-HPSG relies on the Curry-Howard-van Benthem isomorphism (van Benthem 1991), once again the logic is enough and no special Semantics Principle is necessary. Section 3 describes the logic and Section 4 elaborates on the differences between HPSG and TL-HPSG.

(1) $\underbrace{\text{'John'}}_{j}$ *'john'* : e_j

$$\text{where } j = \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{FORM} & \mathit{nform} \\ \text{CASE} & \\ \text{AGR} & \begin{bmatrix} \text{PER} & \mathit{3rd} \\ \text{NUM} & \mathit{sing} \\ \text{GEND} & \mathit{masc} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{ARG-ST} & \langle\rangle \end{bmatrix}$$

(2) $\underbrace{s \text{ 'snores'}}_{f}$ *'*$\lambda x.\, \mathit{snore}(x)$*'* : $e_s \multimap t_f$

$$\text{where } f = \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{FORM} & \mathit{fin} \end{bmatrix} \end{bmatrix} \\ \text{ARG-ST} & \langle\ s\ \rangle \end{bmatrix}$$

$$\text{where } s = \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{FORM} & \mathit{nform} \\ \text{CASE} & \mathit{nom} \\ \text{AGR} & \begin{bmatrix} \text{PER} & \mathit{3rd} \\ \text{NUM} & \mathit{sing} \\ \text{GEND} & \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(3) John snores.

$$\frac{\Gamma, X : T \vdash E : T'}{\Gamma \vdash \lambda X.E : (T \multimap T')} (\multimap Intro.)$$

$$\frac{\Gamma_1 \vdash A_1 : T'_1 \quad \ldots \quad \Gamma_N \vdash A_N : T'_N}{F : T_1 \multimap \ldots \multimap T_{N+1}, \Gamma_1, \ldots, \Gamma_N \vdash F\ A_1\ \ldots\ A_N : T_{N+1[\sigma]}} (\multimap Elim.)$$

$[T_{1[\sigma]}{=}T'_{1[\sigma]}, \ldots, T_{N[\sigma]}{=}T'_{N[\sigma]}$, and T_{N+1} is a base type.]

where σ is some total function from variables to individual denoting expressions such that for any variable V, $\sigma(V) \neq V$.

Figure 1: TL-LFG/TL-HPSG ($G3_i$) Type-Inference Rules

3 The Glue Type-Logical Framework

Glue was originally proposed as the syntax-semantics interface of LFG (Dalrymple, Lamping, and Saraswat 1993), but it was soon realised that it could be used with a variety of different syntactic formalisms: LTAG (Frank and van Genabith 2001), HPSG (Asudeh and Crouch 2002), CG (Asudeh and Crouch 2001), and CFG (Asudeh and Crouch 2001). A new development came from the realisation that it can actually be used directly with the word sequence instead of some elaborate syntactic structure. This section discusses what Glue does and how, making a distinction between the original usage and its usage as the underlying formal foundation for TL-HPSG.

The semantic constructors of a word sequence comprise its typing context Γ. Given Γ and the target type T, Glue is then responsible for deriving all possible meanings M that would have T as their syntax-semantics interface type. If $M_1, \ldots M_n$ are the atomic meanings of the words in a sentence and $T_1, \ldots T_n$ are their syntax-semantics interface types, then what the type-logical setup described ($M_1 : T_1, \ldots M_n : T_n \vdash M : T$) provides is the syntactically licensed function(s) from the meanings of the atomic components to the meaning of the whole: $\lambda M_1. \ldots \lambda M_n. M$.

(4)

$$\textit{`john'} : e_j, \textit{`}\lambda x.\, \textit{snore}(x)\textit{'} : e_s \multimap t_f \vdash \textit{`}\lambda x.\, \textit{snore}(x)\textit{'}\ \textit{`john'} : t_g$$

The difference between traditional uses of Glue and the proposed use of Glue as the formal foundation for HPSG grammars is that in the former case all important syntactic relations can be assumed to have already been established (i.e. $s = j$ and $f = g$), whereas in the later these relations are left to Glue to figure out. The initial picture at the start of a TL-HPSG derivation is that a number of underspecified feature structures are a part of the types and not all of the relations between them are known. This changes gradually during the concurrent syntactic analysis / semantic composition process (or its dual for generation rather than parsing). The derivation in (4) is possible exactly because the unificational First-Order Glue type system used as the formal foundation for TL-HPSG (Figure 1) would allow j to unify with s and f with g. The whole setup is identical to what one would have in the various versions of TL-CG.

4 Meta-theoretical observations

TL-HPSG has a notion of a semantic constructor consisting of a semantic component and a syntax-semantics interface type. The HPSG of Sag et al. (2003) builds the semantics into the feature structure (the SEM component), but falls short of providing full details for meaning composition, treating this as a somewhat complicated matter not appropriate for an introductory book on syntax.[2] Pollard and Sag (1994) stipulated a Semantics Principle which was indeed somewhat complicated. Asudeh and Crouch (2002) criticised that Semantics Principle on the grounds of it needing to distinguish between the semantic contributions of quantifiers and other signs, and also adjunct and non-adjunct daughters, as well as handling quantifier scoping in an ad-hoc manner. They proposed Glue as a suitable syntax-semantics interface for HPSG, but as they tried not to distance themselves from the HPSG tradition of putting semantics into the feature structure, their proposal did not bring out the full elegance of the Glue approach. Instead of embedding Glue in HPSG and replacing the Semantic Principle of Pollard and Sag (1994) with a Glue-based one, TL-HPSG is based on a formal foundation that treats syntax and semantics concurrently, on an equal basis, and with no special provisions. One consequence is that TL-HPSG has no use for the stipulation of a Semantics Principle.

In HPSG, the HEAD feature was helpful (as a way of grouping features) in the stipulation of two HPSG generalisations, the Head Feature Principle ('the HEAD of the mother node equals the HEAD of the head daughter node') and the Valence Principle ('unless stated otherwise, the VAL features of a mother node are identical to those of its head daughter'). In TL-HPSG, just like in CG, there is no distinction between the feature structure of the head and the result of combining the head with its arguments and modifiers. Without this distinction, the Head Feature Principle (5) and the Valence Principle (6) need not be stipulated.

(5) **Head Feature Principle:**

$$\textit{hd-cx}: \begin{bmatrix} \textsc{mother} & \begin{bmatrix} \textsc{syn} & \begin{bmatrix} \textsc{head} & \boxed{1} \end{bmatrix} \end{bmatrix} \\ \textsc{hd-dtr} & \begin{bmatrix} \textsc{syn} & \begin{bmatrix} \textsc{head} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(6) **Valence Principle:**

$$\textit{hd-comp-cx}: \begin{bmatrix} \textsc{mother} & \begin{bmatrix} \textsc{syn} & \begin{bmatrix} \textsc{val} & \boxed{/1} \end{bmatrix} \end{bmatrix} \\ \textsc{hd-dtr} & \begin{bmatrix} \textsc{syn} & \begin{bmatrix} \textsc{val} & \boxed{/1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The HPSG separation of the phrase feature structure and the head constituent's feature structure is important for the Head-Specifier Rule (7) and the Head-Complement Rule (8). Again, these have no place in TL-HPSG. Nor do the COMPS and SPEC features as the information they try to capture is in the structure of the types just like in CG. The Head-Modifier Rule (9) is also not necessary and neither is the MOD feature, for similar reasons.

[2] Ironically, it is by basing HPSG on Glue, a theory of the syntax-semantics interface, that the current work formally simplifies HPSG.

(7) **Head Specifier Rule:**
hd-comp-cx :

$$\begin{bmatrix} \text{MOTHER} & \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{VAL} & \begin{bmatrix} \text{SPR} & \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{HD-DTR} & \boxed{0} \begin{bmatrix} word \\ \text{SYN} & \begin{bmatrix} \text{VAL} & \begin{bmatrix} \text{SPR} & \langle \boxed{1} \rangle \\ \text{COMPS} & \langle\rangle \\ \text{STOP-GAP} & \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} & \langle \boxed{1}, \boxed{0} \rangle \end{bmatrix}$$

(8) **Head Complement Rule:**
hd-comp-cx :

$$\begin{bmatrix} \text{MOTHER} & \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{VAL} & \begin{bmatrix} \text{COMPS} & \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{HD-DTR} & \boxed{0} \begin{bmatrix} word \\ \text{SYN} & \begin{bmatrix} \text{VAL} & \begin{bmatrix} \text{COMPS} & \boxed{A} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} & \langle \boxed{0} \rangle \oplus (\boxed{A} nelist) \end{bmatrix}$$

(9) **Head Modifier Rule:**
hd-mod-cx :

$$\begin{bmatrix} \text{HD-DTR} & \boxed{0} \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{VAL} & \begin{bmatrix} \text{COMPS} & \langle\rangle \\ \text{STOP-GAP} & \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} & \langle \boxed{0}, \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{VAL} & \begin{bmatrix} \text{COMPS} & \langle\rangle \\ \text{MOD} & \langle \boxed{0} \rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \rangle \end{bmatrix}$$

A closer look at how TL-HPSG provides a unified approach to linguistic phenomena on the basic of functors and arguments (i.e. linear implication) extends to coordination and gapping is beyond the scope of this paper. However, it is worth observing that there is nothing special about 'displaced' material as far as the logic is concerned and that coordination is, at its most common form, handled very naturally by a functor taking two similar arguments and returning a similar result. By following the practice of thinking in terms of functors and arguments (common in Montague Grammar, Glue, and Categorial Grammar), TL-HPSG deals with linguistic structures in a much more direct way than by the series of stipulations of fragmentary 'syntactic generalisations' HPSG resorts to.

5 Type Hierarchy

Elaborate type hierarchies are an important feature of large-scale HPSG grammars. HPSG relies on unification when it comes to features obtaining their values. However, it is a prerequisite that the types of the feature structures to be unified are compatible. So an HPSG feature structure may be regarded as a pair (t, f) of a type t which is part of a type hierarchy which defines a subtyping relation and a simple feature structure f where unification is all that matters. If an HPSG feature structure $h = (t, f)$ is to be used where a feature structure $h' = (t', f')$ is expected this can only be if $t \sqsupseteq t'$ (t is a stronger type than t' i.e. a subtype of t') and there is some assignment function σ such that $f[\sigma] = f'[\sigma]$ (f and f' can unify).

Subtyping is an asymmetric relation. Unification-compatibility ("unifies with") is a symmetric one. Without subtyping in Glue, it would seem that TL-HPSG cannot legitimately claim to be a genuine attempt to preserve the intuitions of HPSG. However, while unification is symmetric, there is an asymmetry inherent in the logic. A feature structure can be an argument (what is actually supplied) to a function or it can be a function's parameter (what is expected and consumed). Because unification is commutative ($f_1 \sqcap f_2 = f_2 \sqcap f_1$, assuming f_1 and f_2 are unification-compatible), unification-compatibility is a symmetric relation ($f_1 \mathbb{U} f_2$ if and only if $f_2 \mathbb{U} f_1$). However, derivability is not: $f(U) \vdash f(5)$, while $f(5) \nvdash f(U)$. It is thanks to this asymmetry that type hierarchies can be encoded in First-Order Glue.

Suppose type A has a subtype B, B has a subtype C, and that there is a type D that is unrelated to A, B, and C (Figure 2). A function f with a parameter of type B can, of course, accept arguments of type B, but also of type C; arguments of type A or type D are not acceptable. The challenge is to try to find a way of preserving the intended behaviour of subtyping in a system that does not include it in its formal machinery.

Let each feature structure that would have been considered as having one of the types incorporated in the subtyping hierarchy of Figure 2 have an attribute for each type in that hierarchy. These attributes can have a plus or a minus value or they can be left underspecified.

For argument features, let the value of each type attribute be '-' if the feature structure is not of that type (and consequently not of a stronger one either); let it be underspecified otherwise. So, a parameter feature structure of type B will have its A, and B attributes positive and its C and D attributes underspecified.

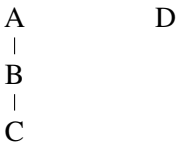

Figure 2: Sample type hierarchy

Argument Compatibility Matrix	(A_{arg}) [A, B, C, D: −]	(B_{arg}) [A, B, C, D: −]	(C_{arg}) [A, B, C, D: −]	(D_{arg}) [A, B, C, D: −]
(A_{par}) [A: +]	✓	✓	✓	
(B_{par}) [A: +, B: +]		✓	✓	
(C_{par}) [A: +, B: +, C: +]			✓	
(D_{par}) [D: +]				✓

Table 1: Argument compatibility—assuming the type hierarchy of Figure 2.

For parameter features, let the value of each type attribute be '+' if the feature structure may be of that type (or a stronger one); let it be underspecified otherwise. An argument feature structure of type A will have its B, C, and D attributes negative and its A attribute underspecified. An argument feature structure of type B will have its C and D attributes negative and its A and B attributes underspecified. An argument feature structure of type C will have its D attribute negative and its A, B,C attributes underspecified. Finally, an argument feature structure of type D will have all of its A, B, and C attributes negative and its D attribute underspecified.

Therefore, only argument feature structures of type B and C can be used as arguments of function f having a parameter of type B. The complete picture is given in Table 5.

This encoding was inspired by an LFG treatment of case (Dalrymple, King, and Sadler 2006) which, unlike the standard account (Dalrymple and Kaplan 2000), does not use sets. The avoidance of LFG sets was the reason this analysis was particularly relevant to TL-LFG (Kokkonidis 2007a; Kokkonidis 2007b). It was the realisation that this LFG analysis for case marking provides the necessary asymmetry for the encoding of type hierarchies that lead to the design of TL-HPSG.

6 Conclusions

To summarise, the design of TL-HPSG offers the simplicity of Glue while remaining close to the key underlying ideas of HPSG; if anything it seems to embody these ideas in a more formally parsimonious and direct manner. It was not the aim of this paper to give a full exposition, but rather a first taste of TL-HPSG. There is ongoing work on coordination, gapping, and anaphoric binding. There is also ongoing work on exploring the consequences of TL-HPSG's decidability, one of its majors differences with HPSG.

Acknowledgements. I am greatly indebted to Mary Dalrymple and Oiwi Parker Jones for helpful comments and suggestions on this paper. I would also like to thank the audience at LingO 2007, but also Doug Arnold, Nissim Francez, Manfred Sailer and Anders Soegaard for their comments on related presentations at the 2nd International Workshop on Typed Feature Structure Grammars and the 16th Amsterdam Colloquium. This work has been supported by the Arts and Humanities Research Council (grant 2005/118667).

References

Asudeh, A. and R. Crouch (2001). Glue semantics: A general theory of meaning composition. Talk given at Stanford Semantics Fest 2, March 16, 2001.

Asudeh, A. and R. Crouch (2002). Glue semantics for HPSG. In F. van Eynde, L. Hellan, and D. Beermann (Eds.), *Proceedings of the 8th International HPSG Conference*. Stanford, CA.: CSLI Publications.

Dalrymple, M. (Ed.) (1999). *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press.

Dalrymple, M. and R. Kaplan (2000). Feature indeterminacy and feature resolution. *Language 76*, 759–798.

Dalrymple, M., T. H. King, and L. Sadler (2006). Indeterminacy by underspecification. In M. Butt and T. King (Eds.), *Proceedings of the Eleventh International LFG Conference*, CSLI Online Publications.

Dalrymple, M., J. Lamping, and V. Saraswat (1993). LFG semantics via constraints. In *Proceedings of the Sixth Meeting of the European ACL*, University of Utrecht, pp. 97–105. European Chapter of the Association for Computational Linguistics.

Frank, A. and J. van Genabith (2001). GlueTag: Linear Logic based semantics construction for LTAG — and what it teaches us about the relation between LFG and LTAG —. In *Proceedings of the LFG01 Conference*. CSLI Publications.

Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science 50*, 1–102.

Kokkonidis, M. (2006). A simple linear first-order system for meaning assembly. In *Proceedings of the Second International Congress on Tools for Teaching Logic*, Salamanca, Spain.

Kokkonidis, M. (2007a). Encoding LFG f-structures in the TL-LFG type system. In *Proceedings of the Second International Workshop on Typed Feature Structure Grammars*, Tartu, Estonia.

Kokkonidis, M. (2007b). Towards a functional type-logical theory of grammar. In M. Butt and T. H. King (Eds.), *Proceedings of the LFG07 Conference*. CSLI Publications. Forthcoming.

Kokkonidis, M. (2008). First-Order Glue. *Journal of Logic, Language and Information 17*, 43–68.

Moortgat, M. (1997). Categorial type logics. In J. van Benthem and A. ter Meulen (Eds.), *Handbook of Logic and Language*. Elsevier.

Morrill, G. V. (1994). *Type Logical Grammar: Categorial Logic of Signs*. Dordrecht: Kluwer.

Pollard, C. (2004). Type-logical HPSG. In G. Jaeger, P. Monachesi, G. Penn, and S. Wintner (Eds.), *Proceedings of Formal Grammar 2004 (Nancy)*, pp. 107–124.

Pollard, C. J. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Sag, I. A., T. Wasow, and E. Bender (2003). *Head-Driven Phrase Structure Grammar: A Formal Introduction* (Second ed.). Chicago: Stanford, CSLI.

van Benthem, J. (1991). *Language in Action*. Amsterdam: North-Holland.